IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s) :	James R. WASON	Group Art Unit: 2176
Appln. No. :	10/606,547	Examiner: Maikhanh Nguyen
Filed :	June 26, 2003	Confirmation No.: 5225
For :	**RICH TEXT HANDLING FOR A WEB APPLICATION**	

United States Patent and Trademark Office
Customer Service Window, Mail Stop Amendment
Randolph Building
401 Dulany Street
Alexandria, VA 22314

APPLICANTS' SEPARATE RECORD OF TELEPHONE INTERVIEW

Sir:

In response to the September 21, 2010 Telephone Interview, please find below Applicants' separate record of the interview. The courtesies extended to Applicants' representative by Examiner Nguyen during the telephone interview are appreciated.

Examiner Nguyen contacted Applicants representative to (1) assert that claims 24-35 and 39-47 are directed to a separate distinct invention; and (2) propose an Examiner's Amendment to place this application in condition for allowance. Specifically regarding claims 24-25 and 39-47, Examiner Nguyen asserted that she would issue a Restriction Requirement if claims 24-35 and 39-47 were not canceled. Applicants representative authorized the below proposed amended claim set in order to: (1) amend claim 1 to incorporate the features of claim 51; (2) amend claims 1, 4, 15, 48 and 52 to correct minor informalities; (3) amend claim 48 to recite statutory subject matter; and (4) cancel claims 24-35 and 39-47, which would have been directed to a non-elected invention, and to place this application in condition for allowance. Examiner Nguyen agreed to enter the proposed amendments by way of Examiner's Amendment.

AMENDMENT TO THE CLAIMS

Please **AMEND** claims 1, 4, 15, 48 and 52 as follows.

Please **CANCEL** claims 24-35, 39-47 and 51 without prejudice or disclaimer.

A copy of all pending claims and a status of the claims are provided below.

1. (Currently Amended) A method of representing and managing rich text for use by Web based applications and browsers as implemented in a machine, the method comprising the steps of:

providing one or more classes for use by the applications to at least create and manage one or more rich text nodes in a memory structure representation representative of rich text;

representing the rich text in the memory structure representation; and

editing the rich text in a document using the memory structure representation to perform editing functions on the document having the rich text as managed and created by the one or more classes, wherein the memory structure representation comprises:

forming a table structure represented in memory as a set of special rich text node types including table node, table body node and table header node for defining table characteristics, wherein the table row node, heading cell node and row cell node correspond to types of html hyper-text markup language (HTML) tags controlling table representation, wherein

each type of node maintains a reference to the nodes it controls for a next level including the table row controlling a list of row cell nodes, and the table body node controlling a list of table row nodes,

the header cell node and row cell node maintain lists of the rich text nodes, representing content of the rich text nodes and the rich text node contains an anchor point to another table node to start a new table at that point in the rich text thereby allowing for nested tables; and further comprising:

providing a method to transform text from its memory format into string representations and vice versa, comprising:

storing one of the rich text as a string in a relational database, formatting the string by converting the rich text into a html HTML string for storage, converting the rich text into xml Extensible Mark-up Language (XML) and using a compressed format where various attributes of each rich text node are captured, along with the text value for that node, wherein creating rich text memory structure from html HTML, comprises one of:

parsing, by the rich text node, a well-formed segment of html HTML and set its attributes accordingly, including creating other rich text nodes as needed as the html HTML indicates a change in text attributes or presence of an image or link; and

as a function in a rich text list, taking the html HTML that is not well formed, and preprocesses the html HTML to make it recognizable by the rich text nodes, wherein the rich text list also handles creating the nodes for the table structures included within the html HTML;

parsing the HTML by extracting tag information from a text attribute, then using the tag information to set other attributes in the rich text by calling a resolveTag method, wherein the resolveTag method comprises:

reading text up to a first tag and if this is not a null string, cloning a current rich text node and making the clone a preceding node, and assigning to it all text before the first tag, then removing the text and calling the resolveTag method again, wherein the HTML is well formed for the cloning to work recursively, and the well formed HTML ensures that encountered tags are in proper order so that the text sent to the clone will not miss any tags;

if the tag has a matching end tag, checking if there is any text beyond the end tag, and if there is, cloning the current rich text node, making the clone the following node,

and assigning it the text after the end tag, then removing the text and calling the resolveTag method again;

if the tag is an image or link tag, cloning the current rich text node and making the clone the following node, and assigning the following node the text after the tag;

passing the tag information to resolve the tag and to set up tag attributes, wherein if there is an image or link tag, the attributes are stored in the text; and

if preceding or following nodes are not null, call resolveTag making the preceding or following node the current node, which recursively propagates more rich text nodes to fully represent the rich text.

2. (Original) The method of claim 1, wherein the providing the one or more classes includes the steps of:

providing a rich text list class for managing the one or more rich text nodes in the memory structure representation;

providing a rich text class to create the one or more rich text nodes each representing a unit of rich text and its attributes; and

instantiating the rich text list class and the rich text class.

3. (Previously Presented) The method of claim l, wherein the representing rich text step includes representing the string representations.

4. (Currently Amended) The method of claim 3, wherein the string representations comprise at least one of a character large object (CLOB), hyper-text markup language (HTML) HTML, extensible markup language (XML) XML, plain text, and spell check text.

5. (Original) The method of claim 1, wherein the providing one or more classes step includes providing rich text attributes, wherein the attributes include at least one of font face, font size, font color, italicized, underlined, and bold.

6. (Original) The method of claim 1, wherein the providing one or more classes step includes providing properties associated with the one or more rich text nodes, the properties comprising at least one of a line break, a table, an image, a link, and text.

7-11. (Canceled)

12. (Previously Presented) The method of claim 1, further comprising the steps of:

providing well-formed segments of text to a current rich text node of the one or more rich text nodes from a rich text list node;

parsing the well-formed segments of text; assigning unparsed segments of text to the current rich text node's text attribute; and

resolving the current rich text node's text attribute by extracting tag in formation and setting attributes in the current rich text node, the attributes including at least one of font face, font size, font color, italicized, underlined, and bold.

13. (Original) The method of claim 12, wherein the providing well-formed segments step comprises the steps of:

suppressing certain tags associated with some the unparsed segments by changing starting and ending tags to substitution strings;

checking whether the starting and ending tags are in proper order and eliminating pairs of the starting and the ending tags that have null content;

converting some of the substitution strings to original values; and

reconstituting the well-formed segments of text into one string when pairs of starting and end tags are eliminated.

14. (Original) The method of claim 12, wherein the providing well-formed segments step comprises the steps of:

restoring table related tags; and

breaking the well-formed segments at table tags and organizing the broken

segments into a new rich text list node with entries of at least one of vectors and string.

15. (Currently Amended) The method of claim 12, wherein the text is at least one of hypertext mark- up language (html) HTML and extensible mark-up language (xml) XML.

16. (Previously Presented) A method of representing and managing rich text for use by applications as implemented in a machine, the method comprising the steps of:

providing one or more classes for use by the applications to at least create and manage one or more rich text nodes in a memory structure representation representative of rich text;

representing the rich text in the memory structure representation; and

editing the rich text in a document using the memory structure representation to perform editing functions on the document having the rich text as managed and created by the one or more classes, further comprising the steps of:

providing well-formed segments of text to a current rich text node of the one or more rich text nodes from a rich text list node;

parsing the well-formed segments of text; assigning unparsed segments of text to the current rich text node's text attribute; and resolving the current rich text node's text attribute by extracting tag information and setting attributes in the current rich text node, the attributes including at least one of font face, font size, font color, italicized, underlined, and bold;

wherein the resolving step comprises the steps of:

a) reading the text attribute up to a first tag;

b) if the reading step produces a non-null string, then cloning the current rich text node to make a preceding rich text node and assigning to it all text before the tag;

c) checking whether the first tag has a matching end tag;

d) if there is a matching end tag, cloning the current rich text node to make a following rich text node and assigning to it any text after the matching end tag, then removing the text after the matching end tag;

e) resolving the information between the first tag and matching end tag to set up attributes in the current rich text node; and

f) repeating steps a) through e) until a null string is produced in step b).

17. (Original) The method of claim 16, further comprising the step of repeating steps a) through f) on one of the preceding rich text node and the following rich text node.

18. (Original) The method of claim 16, further comprising the step of when the first tag is one of an image tag and a link tag in step a), cloning the current rich text node to make the following rich text node and assigning to the following node the text after the first tag, then continuing with step c).

19. (Original) The method of claim 1, further comprising the steps of:

responding to a request for editing a document containing the rich text;

presenting rich text editing controls for editing the document; and

accepting changes to the document using one or more classes including a rich text class and a rich text list class for editing the document.

20. (Original) The method of claim 19, wherein the accepting changes step includes accepting changes to at least one of a table, a link, an image, and text.

21. (Original) The method of claim 19, wherein the responding step further comprises steps of:

responding to a spell checking request;

presenting a spell check panel that displays spelling alternatives to a misspelled word associated with the one or more rich text nodes; and

accepting a spelling substitution.

22. (Original) The method of claim 21, wherein the responding to a spell checking request step includes searching a spelling dictionary to locate one or more words for presentation in the spell check panel.

23. (Original) The method of claim 22, wherein the one or more words in the dictionary each have one or more associated signatures to aid in locating a match for the misspelled word.

24-35. (Canceled)

36. (Previously Presented) A method of representing and managing documents having rich text for use in a machine, the method comprising the steps of:

representing rich text in a memory structure representation;

providing one or more classes for use by the applications to create the memory structure representation, the one or more classes including a rich text list class to create a rich text list node and to manage one or more rich text nodes and a rich text class to create the one or more rich text nodes each representing a unit of the rich text; and

providing well-formed segments of text to the one or more current rich text nodes from a rich text list node to initialize the current rich text nodes for representing rich text in a document,

wherein the providing well-formed segments of text step further comprising the steps of:

parsing the well-formed segments of text;

assigning unparsed segments of text to the current rich text node's text attribute; and

resolving the current rich text node's text attribute by extracting tag information and sets attributes in the current rich text node, the attributes including at least one of font face, font size, font color, italicized, underlined, and bold,

wherein the resolving step comprises the steps of:

a) reading the text attribute up to a first tag;

b) if the reading step produces a non-null string, then cloning the current rich text node to make a preceding rich text node and assigning to it all text before the tag;

c) checking whether the first tag has a matching end tag;

d) if there is a matching end tag, cloning the current rich text node to make a following rich text node and assigning to it any text after the matching end tag, then removing the text after the matching end tag;

e) resolving the information between the first tag and matching end tag to set up attributes in the current node; and

f) repeating steps a) through e) until all a null string is produced in step b).

37. (Original) The method of claim 36, further comprising the step of repeating steps a) through f) on one of the preceding rich text node and the following rich text node.

38. (Original) The method of claim 36, further comprising the step of when the first tag is one of an image tag and a link tag in step a), cloning the current rich text node to make the following rich text node and assigning to the following node the text after the first tag, then continuing with step e).

39 -47. (Canceled)

48. (Currently Amended) A computer program product comprising a non-transitory computer usable readable storage medium having a computer readable program code embodied in the medium, the computer program product includes:

a first computer program code to provide one or more classes for use by Web based applications and browsers to at least create and manage one or more rich text nodes in a memory structure representation representative of rich text;

a second computer program code to represent the rich text in the memory structure representation;

a third computer program code to edit the rich text in a document using the memory structure representation to perform editing functions on the document having rich text as managed and created by the one or more classes; and

at least one further computer program to parse html hyper-text markup language (HTML) by extracting tag information from a text attribute, then using the tag information to set other attributes in the one or more rich text nodes by calling a resolveTag method, wherein the resolveTag method comprises:

reading text up to a first tag and if this is not a null string, cloning a current rich text node and making the clone a preceding node, and assigning to it all text before the first tag, then removing the text and calling the resolvetag method again, wherein the html HTML is well formed for the cloning to work recursively, and the well formed html HTML ensures that encountered tags are in proper order so that the text sent to the clone will not miss any tags;

if the tag has a matching end tag, checking if there is any text beyond the end tag, and if there is, cloning the current rich text node, making the clone the following node, and assigning it the text after the end tag, then removing the text and calling the resolveTag method again;

if the tag is an image or link tag, cloning the current rich text node and making the clone the following node, and assigning the following node the text after the tag;

passing the tag information to resolve the tag and to set up tag attributes, wherein if there is an image or link tag, the attributes are stored in the text; and

if preceding or following nodes are not null, call resolveTag making the preceding or following node the current node, which recursively propagates more rich text nodes to fully represent the rich text.

49. (Original) The computer program product of claim 48, wherein the computer program product further includes:

a fourth computer program code to provide a rich text list class for creating rich text list nodes and for managing the one or more rich text nodes in the memory structure representation;

a fifth computer program code to provide a rich text class to create the one or more rich text nodes each representing a unit of rich text and its attributes; and

a sixth computer program code to instantiate the rich text list class and the rich text class.

50. (Original) The computer program product of claim 49, wherein the computer program product further includes:

a seventh computer program code to provide well-formed segments of text to a current rich text node from a rich text list node;

an eighth computer program code to parse the well-formed segments of text;

a ninth computer program code to assign unparsed segments of text to the current rich text node's text attribute; and

a tenth computer program code to resolve the current rich text node's text attribute by extracting tag information and to set attributes in the current rich text node.

51. (Canceled)

52. (Currently Amended) The method of claim [[51]] 1, further comprising:

buffering, within each segment html HTML, tags that are not of interest by changing start end and end brackets to substitution strings, which includes a table and list related tags, which are ignored and restored later;

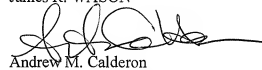
checking to ensure that the tags start and end in the proper order, and each start tag has a matching end tag within the segment, performed by bubbling up end tags that do not have matches within the segment, and then eliminating pairs of start and end tags that have no intervening content;

reconstituting the segments into one string, using a rich text node separator; and breaking the html HTML into segments at tags, and then organizing the segments into a new rich text list that includes entries that are either simple strings for rich text node entries or vectors for table entries.

REMARKS

Claims 1-6, 12-23, 36-38, 48-50 and 52 are currently pending in the application. By the proposed Examiner's Amendment, claims 1, 4, 15, 48 and 52 are amended and claims 24-35, 39-47 and 51 are canceled. Applicants are not conceding in this application that those claims are not patentable over the art cited by the Examiner, as the present claim amendments are only for facilitating expeditious prosecution of the allowable subject matter noted by the examiner. Applicants respectfully reserve the right to pursue the original claims and other claims in one or more continuations and/or divisional patent applications.

Respectfully submitted,
James R. WASON



Andrew M. Calderon
Registration No. 38,093

Roberts, Mlotkowski, Safran & Cole PC
P.O. Box 10064
McLean, VA 22102
Phone: 703.677.3011
Fax: 703.848.2981
Email: acalderon@rmsclaw.com